Consumer and Corporate Affairs Canada | Consommation et Corporations Canada

(11) (A) No. 1 200 911

(45) ISSUED 860218

(52) CLASS 354-230.8

(51) INT. CL.[4] G06F 9/00,15/00

(19) (CA) **CANADIAN PATENT** (12)

(54) System for Automatically Releasing a Dead Lock State in a Data Processing System

(72) Kikuchi, Royoichi,
Japan

(73) Granted to Fujitsu Limited
Japan

(21) APPLICATION No. 310,026

(22) FILED 780824

(30) PRIORITY DATE Japan (110238/77) 770913

No. OF CLAIMS 11

Canada

ABSTRACT OF THE DISCLOSURE

Disclosed herein is a system for automatically releasing a dead lock state in a data processing system, wherein a plurality of kinds of tasks selectively use a plurality of common resources. When one task X occupies a resource A and, in this state, the task is to occupy a resource B, if the resource B is occupied by another task Y, the task X is placed in a waiting state. When the task X, is placed in a waiting state based on the occupation of the resource B by the task Y and the task Y, which is to occupy the resource A, is placed in a waiting state based on the occupation of the resource A by the task X, the task X and the task Y are placed in a dead lock state. When the dead lock state is caused between the tasks X and Y, the occupation of the resource A by the task X is released and the processing of the task Y is carried out, and next, the processing of the task X is carried out.

SYSTEM FOR AUTOMATICALLY RELEASING A DEAD LOCK STATE IN A DATA PROCESSING SYSTEM

FIELD OF THE INVENTION

The present invention relates to a method for automatically releasing a dead lock state in a data processing system wherein a plurality kinds of tasks commonly utilize a plurality of resources.

BACKGROUND OF THE INVENTION

In the data processing system, especially in on-line data processing system, a plurality of kinds of tasks commonly use a plurality of resources (hereinafter, sometimes called "blocks") in carrying out the data processing operation. In the conventional system, when one task is to use one resource, an occupation designation is specified for said one resource by a conventional macro instruction such as "LOCK"or "ENQ" and the utilization of said one resource by another task is inhibited. However, in such a case a so-called dead lock state may be caused for the following reason. That is, when one task X occupies or uses a resource A and, in this state, the task X is to occupy a resource B, if the resource B is occupied by another task Y, the task X is placed in a waiting state.

On the other hand, when the task Y occupies the resource B and, in this state, the task Y is to occupy the resource A, the task Y is also placed in a waiting state. This phenomenon is called a "dead lock state".

Obviously the programming of the data processing system should be planned carefully and the debugging should be carried out so as to avoid the dead lock phenomenon. However, when the number of tasks which are to be processed in parallel increases, the dead lock phenomenon cannot be avoided, even if the programming of the data porcessing system is carefully planned and the debugging is thoroughly carried out.

OBJECTS AND SUMMARY OF THE INVENTION

An object of the present invention is to provide a system which can automatically eliminate a dead lock state in a data processing system.

Another object of the present invention it to provide a system wherein the programming of the data processing apparatus without causing problems which typically occur when programming is planned.

The above-mentioned objects can be achieved by a system for automatically releasing a dead lock state on a data processing system, wherein a plurality of kinds of tasks commonly use a plurality of resources, the system comprising a storage unit having a waiting task control table storing portion which stores the information concerning any task which is in a waiting state due to the occupation of a desired resource by a certain other task,

and a storing before image data buffer portion which stores before image data every time the content of said desired resource is modified. Then when the waiting state is generated with respect to a first task, said system examines the waiting state of the other tasks in accordance with the content of said waiting task control table storing portion and judges whether or not the waiting state of said other tasks is due to the occupation of said resource by said first task. When said waiting state of said other tasks is due to the occupation of said resource by said first task, said system releases the occupation of said resource by said first task, said system carries out the processing of said other tasks in accordance with the content of said before image data buffer portion, and then said system carries out the processing of said first task.

Further features and advantages of the present invention will be apparent from the ensuing description with reference to the accompanying drawings, to which, however, the scope of the invention is in no way limited.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram utilized to describe a dead lock state between two tasks;

Fig. 2 is a block diagram utilized to describe a system which can release the dead lock state between two tasks;

Fig. 3 is a block diagram of one embodiment of the present invention;

Fig. 4 is a block diagram utilized to describe a

system which can release the dead lock state between for tasks;

Fig. 5 is a block diagram of another embodiment of the present invention;

Figs. 6 and 7 are flow charts illustrating the embodiment of Fig. 5

DETAILED EXPLANATION OF THE INVENTION

Referring to Fig. 1, 1X designates a program for carrying out a task X, 1Y designates a program for carrying out a task Y, and 2X and 2Y designate resources. For the purpose of explanation, it is assumed that the task X occupies the resource 2X at a time T_1 by an exclusive macro command, "ENQ A" (for example) and, in this state, at a time T_3 , the task X is to occupy the resource 2Y by the macro command "ENQ B". On the other hand, it is assumed that the task Y occupies the resource 2Y at a time T_2 by the macro command "ENQ B" and, in this state, at a time T_4 , the task Y is to occupy the resource 2X by the macro command "ENQ A". Under these circumstances, at the time the task X generates the command "ENQ B", the task X is place in a waiting state by the task Y and, at the time the task Y generates the command "ENQ A", the task Y is placed in a waiting state by the task X. That is, the tasks X and Y are placed in the dead lock state.

Fig. 2 is a diagram utilized to describe a system which can release a dead lock state between two tasks X

and Y. Referring to Fig. 2, 1X and 1Y designate programs for carrying out the tasks X and Y, respectively, 3X and 3Y designate before image data buffers, respectively, and 4 designates the waiting task control table. In the programs 1X and 1Y shown in Fig. 2, at time (1), the task X occupies and reads a block A; at time (2), the task Y occupies and reads a block C; at time (3), task X modifies the content of the block A; at time (4), the task X occupies and reads of block B; at time (5), the task Y modifies the content of the block C; at time (6), the task X modifies the content of the block B; at time (7), the task X is to occupy the block C, however, it is placed in a waiting state; at time (8), the task Y occupies and reads the block D; at time (9), the task Y modifies the content of the block D; and, at time (10), the task Y is to occupy the block A, however, it is placed in a waiting state.

In the above-mentioned processing, at the time (1), the information "Block A is occupies by task X" is written in the waiting task control table 4; at the time (2), the information "Block C is occupies by task Y" is written in the waiting task control table 4; at the time (3), the before image data of the block A is stored in the before image data buffer 3X; at the time (4), the information "Block B is occupies by task X" is written in the waiting task control table 4; at the time (5), the before image

data of the block C is stored in the before image data buffer 3Y; at the time (6), the before image data of the block B is stored in the before image data buffer 3X; at the time (7), the information "the occupation of the block C by the task X is being kept waiting by the task Y" is written in the waiting task control table 4; at the time (8), the information "Block D is occupied by task Y" is written in the waiting task control table 4; at the time (9), the before image data of the block D is stored in the before image data buffer 3Y; and, at the time (10), the information "the occupation of the block A by the task Y is being kept waiting by the task X" is written in the waiting task control table 4. At this juncture, whether or not the waiting state of the task A is a dead lock state is examined. In the case of Fig. 2, the task X and the task Y are in the dead lock state. Therefore, accordance with the present invention, the blocks C and D, which are occupied by the task Y, are released, but the processing operation of task X is delayed. At this time, the data of the block D, which was modified at the time (9), and the data of the block C, which was modified at the time (5), are returned to their values before modification.

The task X is then released from the waiting state which was caused at time (7), the task X occupies the block C and writes the information "Block C is occupied by task X" in the waiting task control table, and the processing of the block C is carried out by task X. On the other hand, the task Y awaits the release of the occupation

of the block C by the tase X. However, the processing of the block C by the task Y is delayed.

Fig. 3 is a block diagram of one embodiment of the present invention. Referring to Fig. 3, reference numerals 2A, 2B, 2C and 2D designate resources, 3X and 3Y designate before image data buffers, 5 designates a central processing unit (CPV), and 6 designates a main memory MEM, 7 designates a key table, 8-1, 8-2, ... designate a holding queue table, respectively, 9X and 9Y designate a registration table for tasks, respectively. The key table 7 and the registration tables 9X, 9Y in Fig. 3 correspond to the waiting task control table 4 in Fig. 2. Referring to Fig. 3, keys KA, KB ... are provided corresponding to resources 2A, 2B, 2C, ..., respectively, and are used for examining whether or not the resources are occupies when the resources are specified by tasks.

Assuming that the task X and the task Y are programmed as shown in Fig. 2, the data processing according to the present invention is carried out as follows.

(1) When the task X occupies the resource 2A at the time (1), the information designating "the task X" is written at an address position TA (corresponding to the key KA) of the key table 7, in accordance with the key KA corresponding to the resource 2A.

(2) When the task Y occupies the resource 2C at the time (2), the information designating "the task Y" is written at an address position TC (corresponding to the key KC) of the key table 7, in a manner similar to the

process described in item (1), above.

(3) When the task X modifies the content of the resource 2A at the time (3), the before image data of the resource 2A (BLOCK A) is written and stored in one address, for example, a first address, in the before image data buffer 3X.

(4) When the task X occupies the resource 2B at the time (4), the information designating "the task X" is written at an address position TB (corresponding to the key KB) of the table, in a manner similar to the process described in item (1), above.

(5) When the task Y modifies the content of the resource 2C at the time (5), the before image data of the resource 2C (BLOCK C) is written and stored in one address, for example, a first address, in the before image data buffer 3Y.

(6) When the task X modifies the content of the resource 2B at the time (6), the before image data of the resource 2B (BLOCK B) is written and stored in one address, for example, a second address, in the before image data buffer 3X.

(7) When the task X occupy the resource 2C at the time (7), the information "the resource 2C is occupied" as stored in the key KC is detected. Therefore, the information concerning "the task X" is stored in one of the holding queue table 8, such as the table 8-1, and a heading address information A1 in the holding queue table 8 is written in an address portion of location TC (corresponding

to the key KC) in the key table 7.

(8) At this time, the information "the resource 2C is occupied by the task Y" is recognized and, at the same time, in accordance with the content of the address TC in the key table 7, the content of a holding task pointer, in the registration table for tasks 9Y is examined. The content of the holding task pointer indicates whether or not the task Y is being kept waiting by a certain other task, and thus, whether or not the task Y is in the waiting state.

(9) In this example, the task Y is not in the waiting state at the time (7) and, therefore, it is recognized that the task X is in the waiting state. However, it is not in a dead lock state. Therefore, the information of the task Y is written in the holding task pointer in the registration table for tasks 9X, so as to indicate that the task X is in the waiting state because of the task Y. Further, the information of the task X is written in the waiting task pointer in the registration table for tasks 9Y so as to indicate that the task Y is keeping task X waiting.

(10) When the task Y occupies the resource 2D, at the time (8), the information indicating "the task Y" is written in an address portion of location TD (corresponding to the key KD) of the key table 7.

(11) When the task Y modifies the content of the resource 2D at the time (9), the before image data of the resource 2D (BLOCK D) is written and stored in the next (second) address of the before image data buffer 3Y for

storing the before image data of the resources.

(12) When the task Y is to occupy the resource 2A at the time (10), it is recognized by means of the key KA that the resource 2A is already occupied by the task X. Due to this fact, the content of the registration table for tasks 9X is examined so that the content of the holding task pointer, that is, "task Y", is read out, and it is recognized that the task Y is in the waiting state, and also, in the dead lock state at the time (10). Actually, the other registration tables for tasks are examined sequentially, by examining the content of the holding task pointer in one registration table for tasks 9X, and when "the task Y" is written in any one of the registration tables for tasks, it is recognized that the tasks X and Y are in the dead lock state.

(13) When the dead lock state is recognized, the contents of the resources 2D and 2C are restored in accordance with the content of the before image data buffer 3Y and the occupation of the resources 2C and 2D by the task Y is released. That is, the keys KC and KD in the key table are cleared and the corresponding information in locations TC and TD in key table 7 are released. Next, the content of the holding queue table 8-1 is written in the address portion of location TC (corresponding to the key KC) in the key table 7, and the resource 2C is occupied by the task X.

(14) At this time, the content of the before image data buffer 3Y is cleared and the contents of the

registration tables for tasks 9X and 9Y are modified.

As mentioned above, according to the present invention, when a dead lock state is caused, the dead lock state is detected automatically and the occupation of the resource by the task which causes the dead lock state is released. However, the data processing by the tasks is still delayed until before image data of the resource is restored. Thus, the processing program can be planned and developed without imposing the condition of exclusive control of resources and without being overly concerned about the dead lock phenomenon.

Fig. 4 is a diagram of a system which can release a dead lock between four tasks X, Y, Z and α which commonly use resources A, B and C. In (A) of Fig. 4, when the task α is to occupy the resource A or B, at the time T_7 , a dead lock state is caused.

When the task α is to occupy, for example, the resource A and a waiting state is caused, detection of whether or not the dead lock state is caused can be carried out by the following method. That is, the occupation and the waiting are repeated as shown in (A) of Fig. 4 and the waiting task table is formed as shown in (B) of Fig. 4.

Referring to (A) of Fig. 4, the task X is kept waiting by the task Y at time T_3; the task Y also keeps the task Z waiting at time T_4; the task Y is kept waiting by the task α at time T_6. In this condition, when the task α is being kept waiting by the task X at time T_7 a determination of whether or not the task X is in the waiting state is judged

in the left vertical column of (B) of Fig. 4. When the task X is in the waiting state, whether or not the task α is keeping the task X waiting is judged in the left vertical column of (B) of Fig. 4 in the direction of the arrow. If the task X is kept waiting by the task α, the task X and the task α are in a dead lock state.

Fig. 5 is a diagram which shows the automatic release of the dead lock state between the task X and the task α shown in (A) of Fig. 4. Referring to Fig. 5, task controllers 11X, 11Y, 11Z, 11α, resource controllers 12X, 12Y, 12Z, 12α, and isolation wait control tables (IWCT) 13X, 13Y, 13Z, 13α are provided for a plurality of the tasks X, Y, Z and α. The task controllers have the function of controlling the execution of the task, and the resource controllers have the function of controlling the block of the resource occupied by the task.

The isolation wait control tables (IWCT) control all of the waiting states of the tasks. The isolation wait control tables include at least a head pointer, a tail pointer, a next pointer and a parent pointer. The head pointer designates an entry address of the isolation wait control table of the task which was placed in the waiting state prior to any other task by the task which corresponds to said isolation wait control table. The tail pointer designates an entry address of the isolation wait control table of the task which was placed in the waiting state most recently time by the task which corresponds to said isolaiton wait control table. The next pointer designates

an entry address of the isolation wait control table of the task which was placed in the waiting state by the same holding task after the task which corresponds to the isolation control table when said task is in the waiting state. The parent pointer designates an entry address of the isolation wait control table of the holding task when the task which corresponds to the isolation wait control table is kept in the waiting state.

Fig. 5 shows, with regard to Fig. 4, the manner of setting the states of the isolation wait control tables of four tasks X, Y, Z, α as time proceeds. The contents of each isolation wait control table are cleared to "0" state at the initial state.

When the task X is to occupy the block B at the time T_3 , the task X is placed in the waiting state, because the block B is already occupied by the task Y. Therefore, the information "Y" is written in the parent pointer of the isolation wait control table corresponding to the task X. At the same time, the contents of the head pointer and the tail pointer of the isolation wait control table corresponding to the task Y are changed. As only the task Y is keeping the task X waiting at this time, the same information "X" is written in the head pointer and the tail pointer.

When the task Z is to occupy the block B at the time T_4 , the task Z is placed in the waiting state identical to the task X, because the block B is already occupied by the task Y. Therefore, the information "Y" is written in the

parent pointer of the isolation control table corresponding to the task Z. As the task Y is keeping two tasks, that is, the task X and the task Z, waiting, the content of the tail pointer of the isolation control table corresponding to the task Y is changed from "X" to "Z" and, further, the information "Z" is written in the next pointer of the isolation wait control table corresponding to the task X.

When the task Y is to occupy the block C at the time T_6 , the task Y is placed in the waiting state, because the block C is occupied by the task α. Therefore, the information "α" is written in the parent pointer of the isolation wait control table corresponding to the task Y and, at the same time, the contents of the head pointer and the tail pointer corresponding to the task α are changed. As only the task α is keeping the task Y waiting, the same information "Y" is written in the head pointer and the tail pointer 13α.

When one task is to occupy one block, the resource controllers of all other tasks are examined to determine whether or not the block is occupied by another task. When the block is occupied by another task, the parent pointers of the isolation control tables of the other tasks are examined to determine whether the dead lock state has been generated or not. Only when it is determined that the dead lock state has not been generated, is the process of changing the content of the isolation wait control table at the times T_3 , T_4 , T_6 , mentioned above carried out.

Next, the process of judging the dead lock state when

the task α is to occupy the block A at the time T_7 will be explained. When the task α is to occupy the block A at the time T_7 , the task α determines whether or not the block A is occupied by another task, by examining the resource controllers of all other tasks. As the block A is already occupied by the task X at the time T_1 , the block A is already registered in the resource controller corresponding to the task X. Therefore, the task α recognizes that the block A is occupied by the task X and, then, the task α examines the parent pointer of the isolation control table corresponding to the task X. When the fact that the information "Y" is written in the parent pointer of the isolation control table of the task X is recognized, the parent pointer of the isolation control table corresponding to the task Y is examined. As the information "α" is written in the parent pointer of the isolation control table of the task Y, the task α recognizes that, if the task α occupies the block A, a dead lock state will be caused. After the dead lock state is determined in this manner, the task α releases the block C which was occupied by the task α. At the same time the task Y recognizes that the task α has been keeping the task Y waiting, by the contents of the head pointer of isolation wait control table 13α and the tail pointer, and the processing of block C is transferred to the task Y. Therefore, the generation of the dead lock is prevented, the task Y occupies the block C after the time T_8 , not shown in the drawing, and the processing of the task Y is completed.

When the processing of the task Y is finished, the task Y releases the occupation of blocks B and C. Therefore, the processing of these block by the task X and the task α, respectively, is possible. As mentioned above, the dead lock state can be prevented.

Figs. 6 and 7 are flow charts corresponding to the embodiment of Fig. 5.

The key table 7 and the holding queue table 8, shown in Fig. 3, correspond to the resource controllers 12X, 12Y, 12Z, 12α, shown in Fig. 5, the holding task pointers in the registration table for tasks 9X and 9Y, shown in Fig. 3, correspond to the parent pointer and the next pointer, respectively, shown in Fig. 5, and the waiting task pointers in the registration table for tasks 9X and 9Y, shown in Fig. 3, correspond to the head pointer and the tail pointer, respectively, shown in Fig. 5.

Numerous modification and adaptations of the system of the invention will be apparent to those skilled in the art and thus it is intended by the appended claims to cover all such modifications and adaptations which fall within the true spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A system for automatically releasing a dead lock state in a data processing system, wherein a plurality of tasks including a first task and other tasks commonly use a plurality of resources, comprising:

a waiting task control table storing means, one for each given task, for storing information corresponding to said each given task in a waiting state due to occupation of a certain one of said reasources by a certain one of said other tasks, and

a storing before image data buffer means, one for each given task, for storing before image data every time the content of one of said reasources is modified by said each given task, and

said system including examining means, operatively connected to said waiting task control table storing means, responsive to said each given task in the waiting state for examining the waiting states of the other tasks in accordance with the contents of said waiting task control table storing means corresponding thereto, said examining means judging whether or not the waiting state of said other tasks is due to the occupation of said resource by said each given task, and

said system including releasing means responsive to said waiting state of said other tasks due to occupation of said resource by said each given task for releasing the occupation of said resource by said each given task, wherein the processing of said other tasks in

B

accordance with the content of said before image data buffer means occurs prior to processing said first task.

2. The system for automatically releasing a dead lock state in a data processing system according to claim 1, wherein said waiting task control table storing means includes a holding task pointer for recognizing the certain one of said resources and for storing information indicating the certain one of said other tasks by which said each given task is kept waiting, and a waiting task pointer for storing information for controlling the task which is kept waiting by said each given task.

3. The system for automatically releasing a dead lock state in a data processing system according to claim 2, said waiting task control table storing means including key table means, one for each one of said plurality of resources, for holding information indicating occupation of said each one of said plurality of resources by a given one said tasks.

4. The system for automatically releasing a dead lock state in a data processing system according to claim 1, said waiting task control table storing means including key table means, one for each one of said plurality of resources, for holding information indicating occupation of said each one of said plurality of resources by a given one of said tasks.

5. The system for automatically releasing a dead lock state in a data processing system according to claim 3, further comprising holding queue means for holding task information corresponding to said each given task which is

in a waiting state.

6. The system for automatically releasing a dead lock state in a data processing system according to claim 3, further comprising holding queue means for holding task information corresponding to said each given task which is in a waiting state.

7. The system for automatically releasing a dead lock state in a data processing system according to cliam 2, further comprising holding queue means for holding task information corresponding to said each given task which is in a waiting state.

8. The system for automatically releasing a dead lock state in a data processing system according to claim 1, wherein each said waiting task control table storing means includes a head pointer portion for holding an entry address, in said waiting task control table storing means, of a first one of said each given task waiting for said corresponding one of said other tasks.

9. The system for automatically releasing a dead lock state in a data processing system according to claim 1, wherein each said waiting task cotnrol table storing means includes a tail pointer portion for holding an entry address, in said waiting task control table storing means, of the most recent one of said tasks waiting for said corresponding one of said other tasks.

10. The system for automatically releasing a dead lock state in a data processing system according to claim 1, wherein each said waiting task control table storing means

includes a next pointer portion for controlling all waiting states of each of said each given task.

11. The system for automatically releasing a dead lock state in a data processing system according to claim 1, wherein each said waiting task control table storing means includes a parent pointer portion for holding an entry address, in said waiting task control table storing means, of each one of said certain other tasks for which said each given task in a waiting state is waiting.

Fig. 1

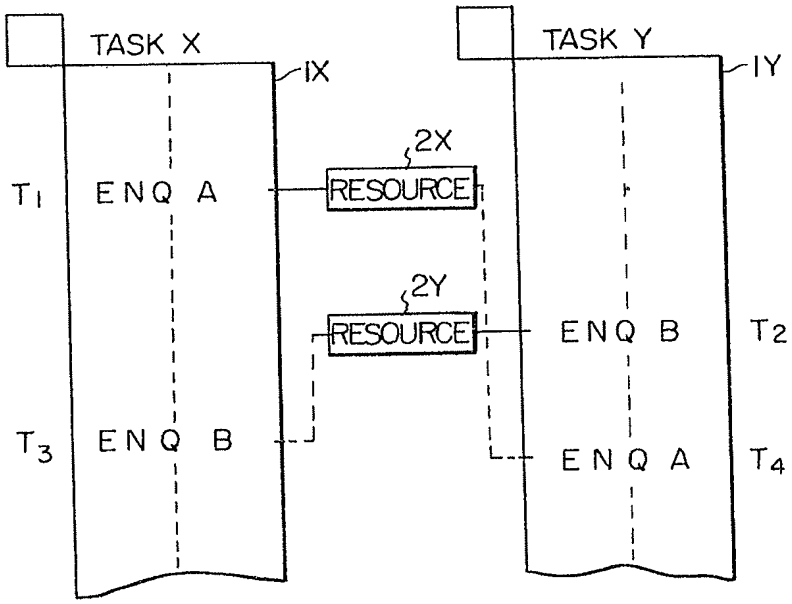
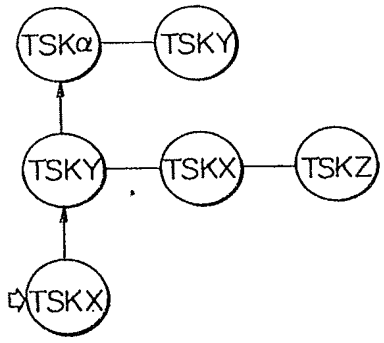


Fig. 4

(A)

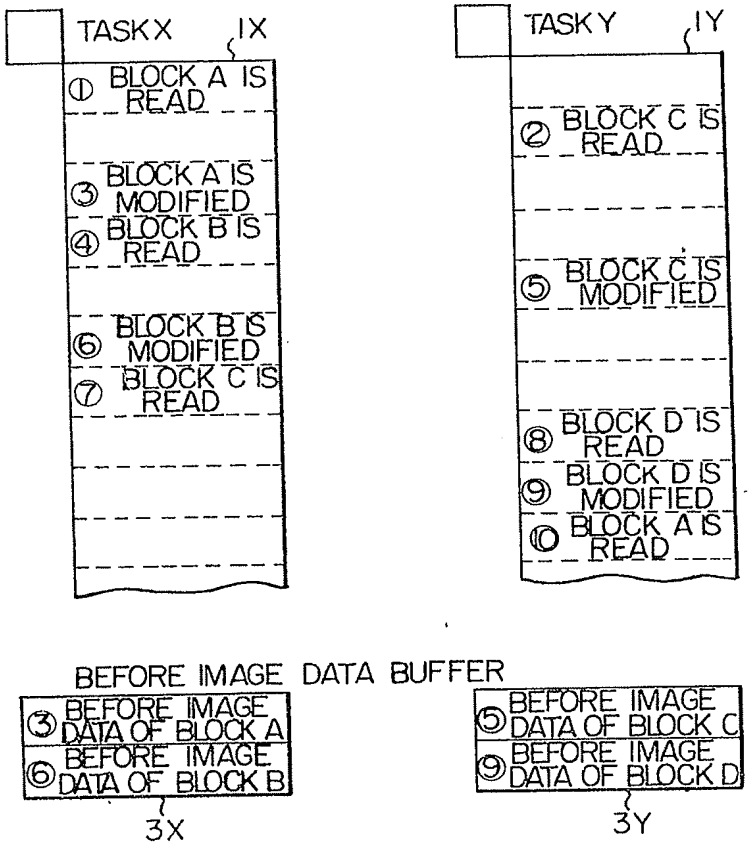
	TSK X	TSK Y	TSKZ	TSK α
T1	A (OCCUPIED)			
T2		B (OCCUPIED)		
T3	B (WAITED)			
T4			B (WAITED)	
T5				C (OCCUPIED)
T6		C (WAITED)		
T7				Ⓐ Ⓑ

(B)



McFadden, Fincham & Co.
Patent Agents

Fig. 2



4

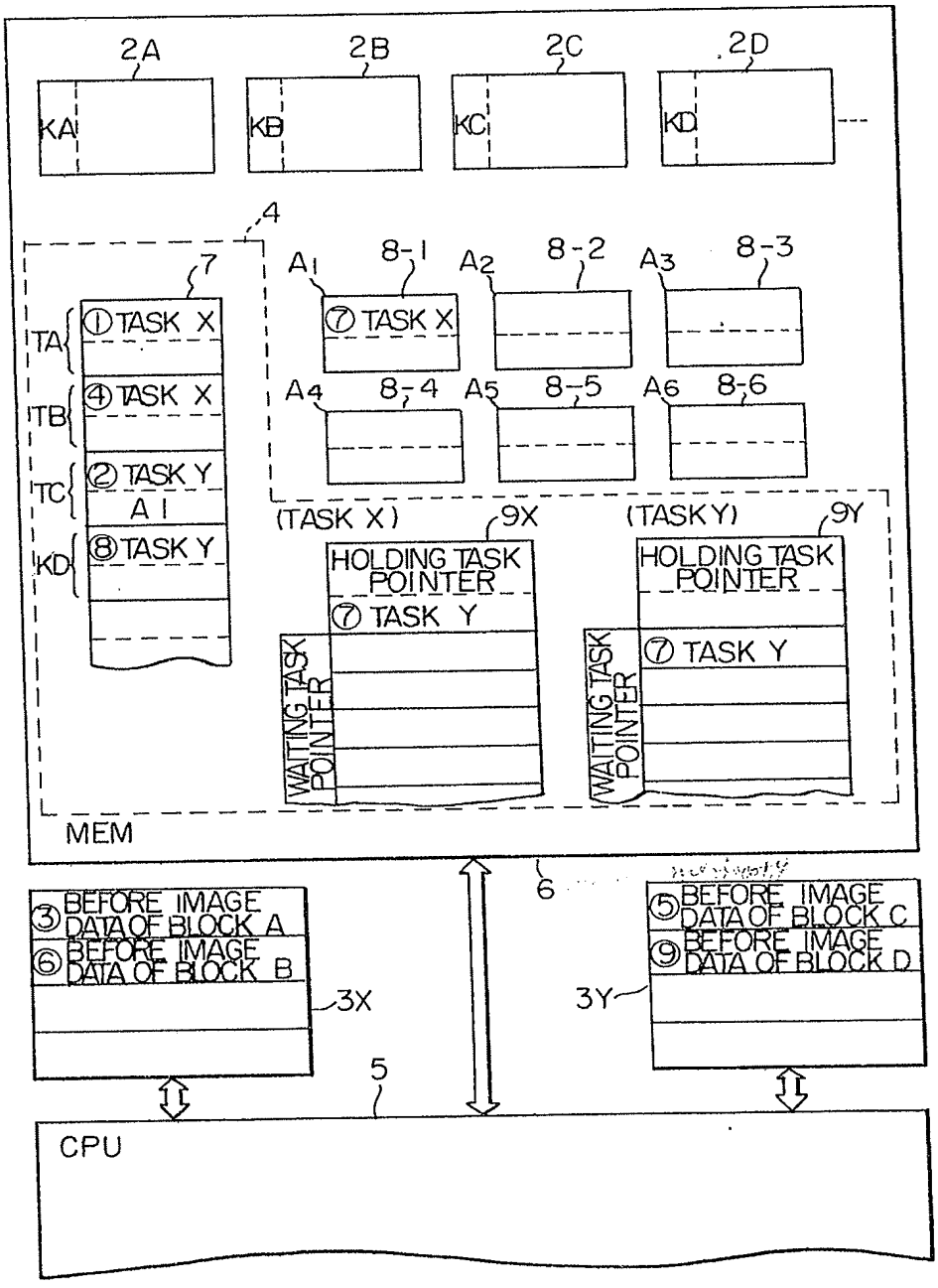
TIMING	①	②	④	⑦	⑧	⑩
X	A (OCCUPIED)		B (OCCUPIED)	C (WAITED BY TASK Y)		
Y		C (OCCUPIED)			D (OCCUPIED)	A (WAITED BY TASK X)

DEAD LOCK

WAITING TASK CONTROL TABLE

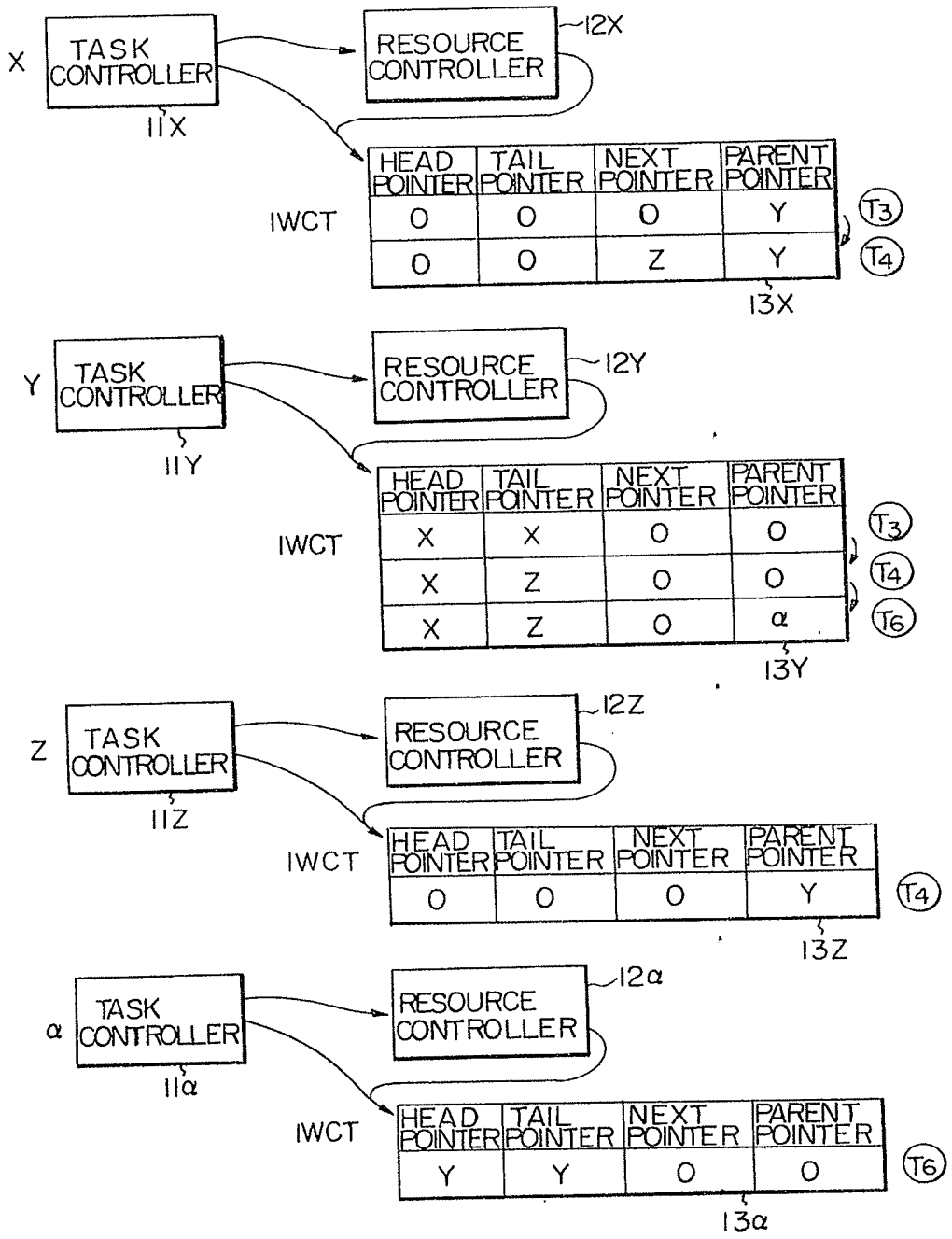
McLadden Jordan & Co.
Patent Agents

Fig. 3



McFadden, Fincham & Co.
Patent Agents

Fig. 5

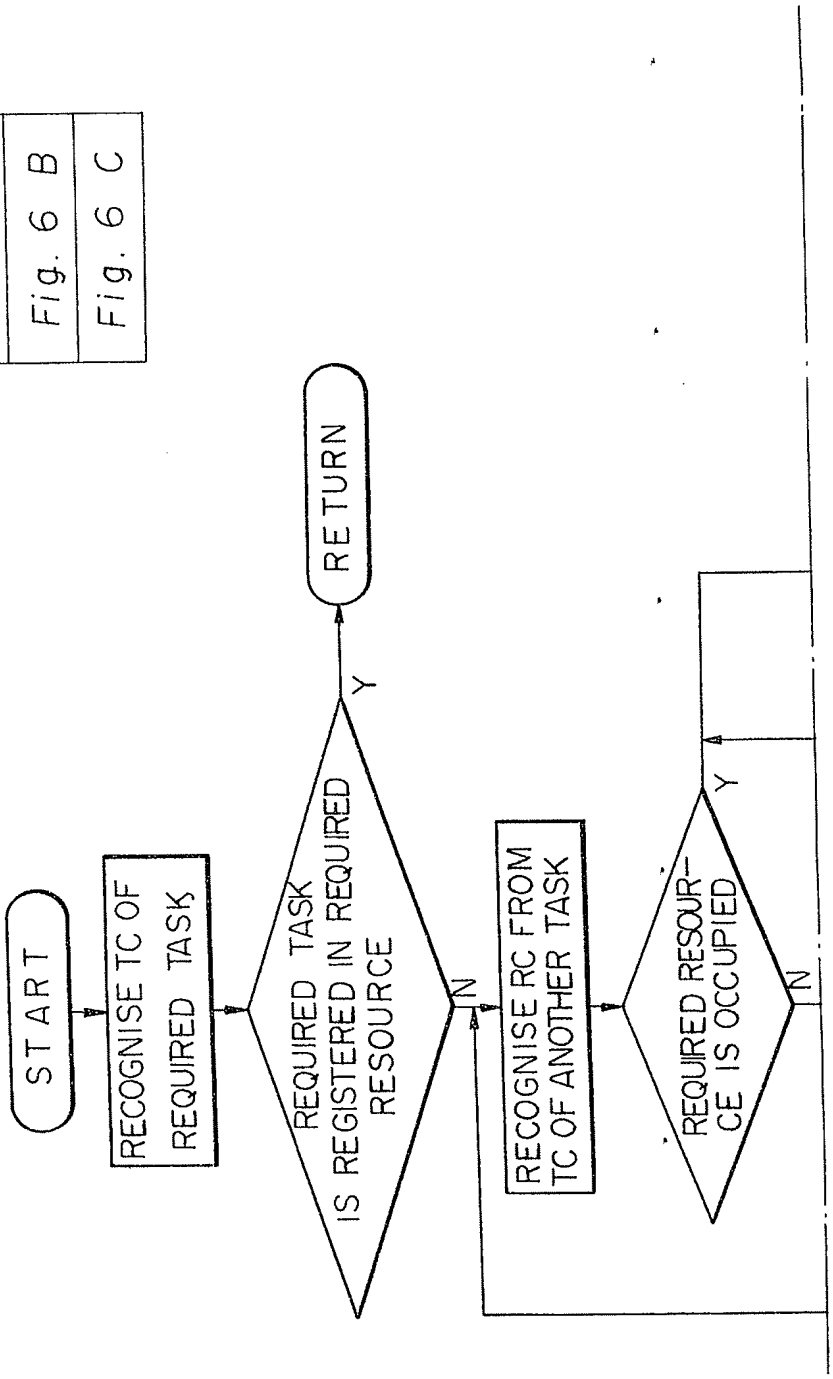


McFadden, Fincham & Co.
Patent Agents

Fig. 6

Fig. 6 A
Fig. 6 B
Fig. 6 C

Fig. 6A



McFadden Fincham & Co.
Patent Agents

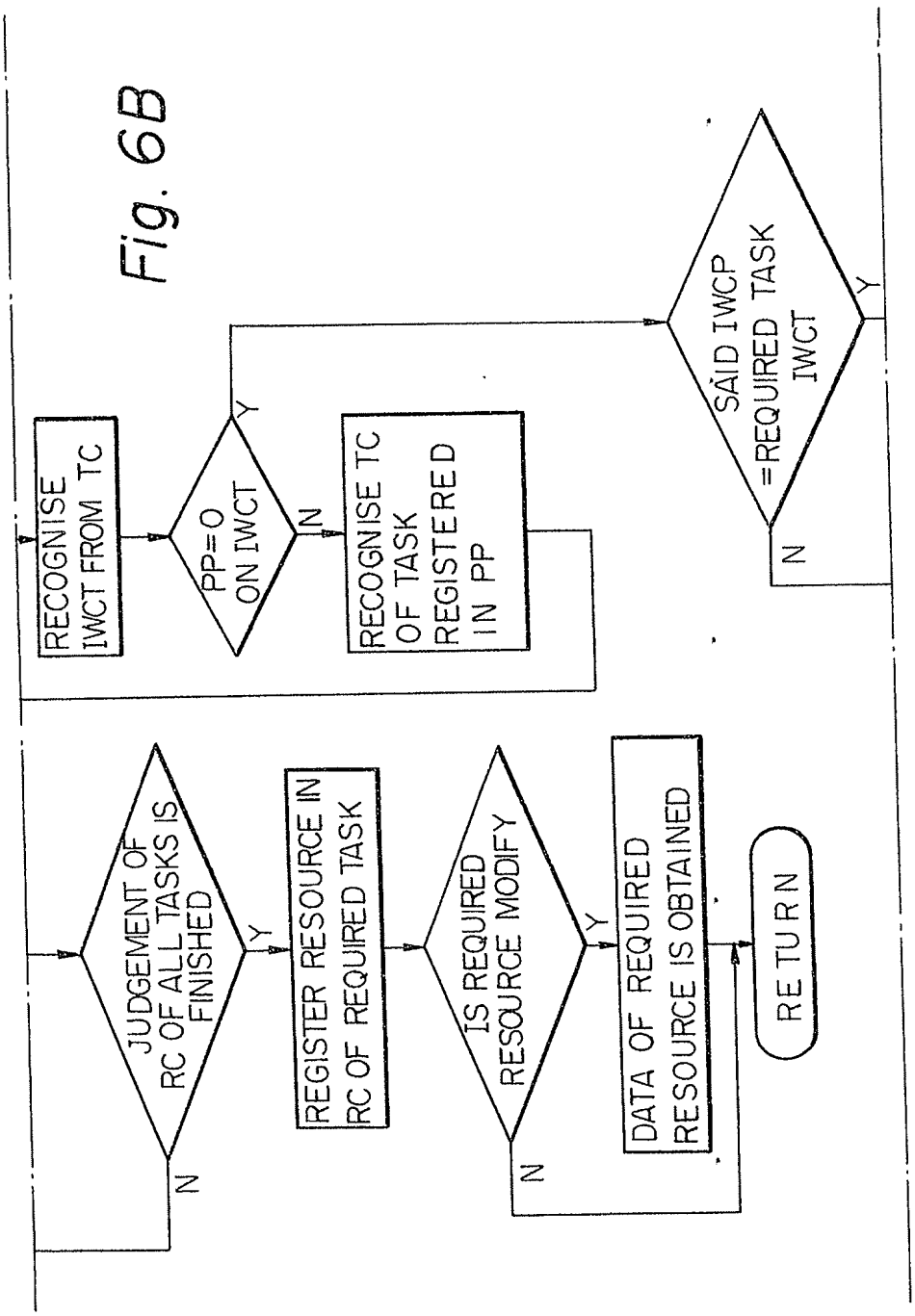


Fig. 6B

Mr Fadden Fincham & Co.
Patent Agents

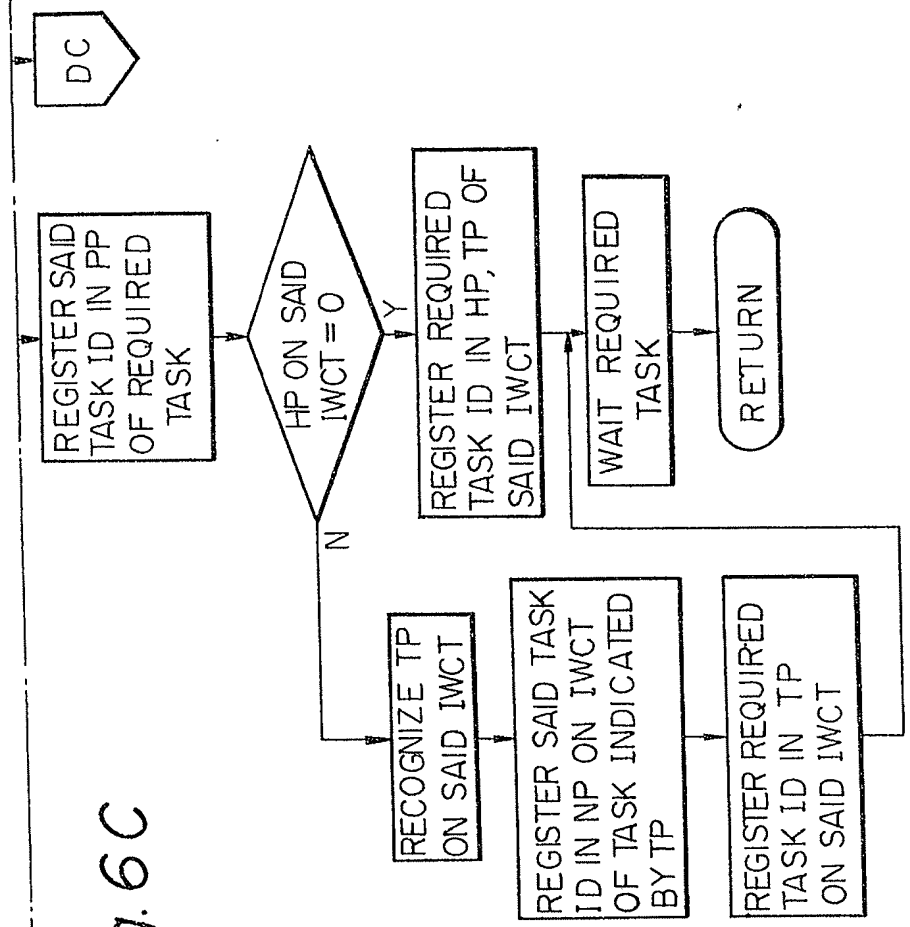
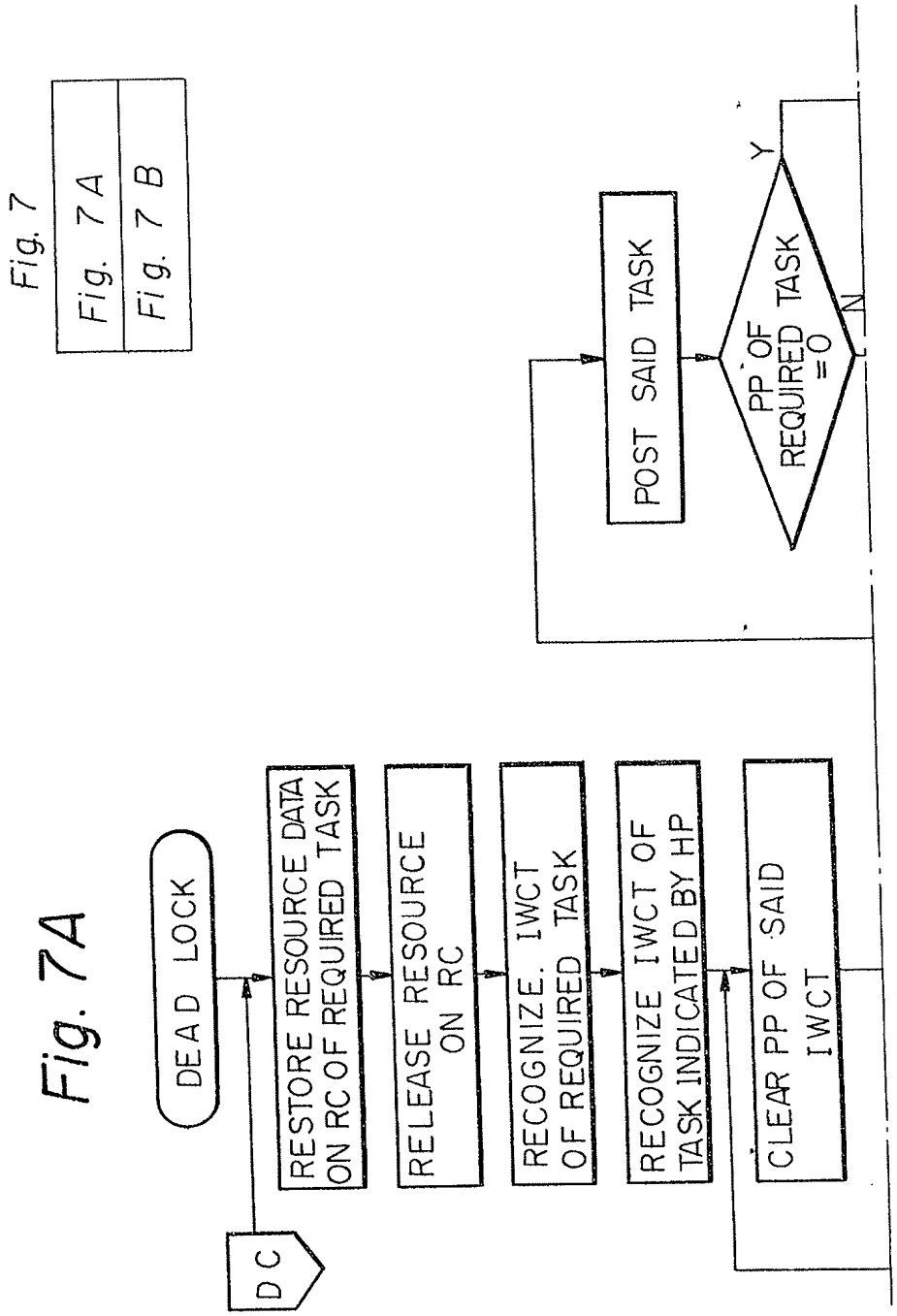


Fig. 6C

Mathys Squire Fricker & Co.
Patent Agents

Fig. 7

Fig. 7A
Fig. 7B



Mk Faclden Freeham + C.
Patent Agents

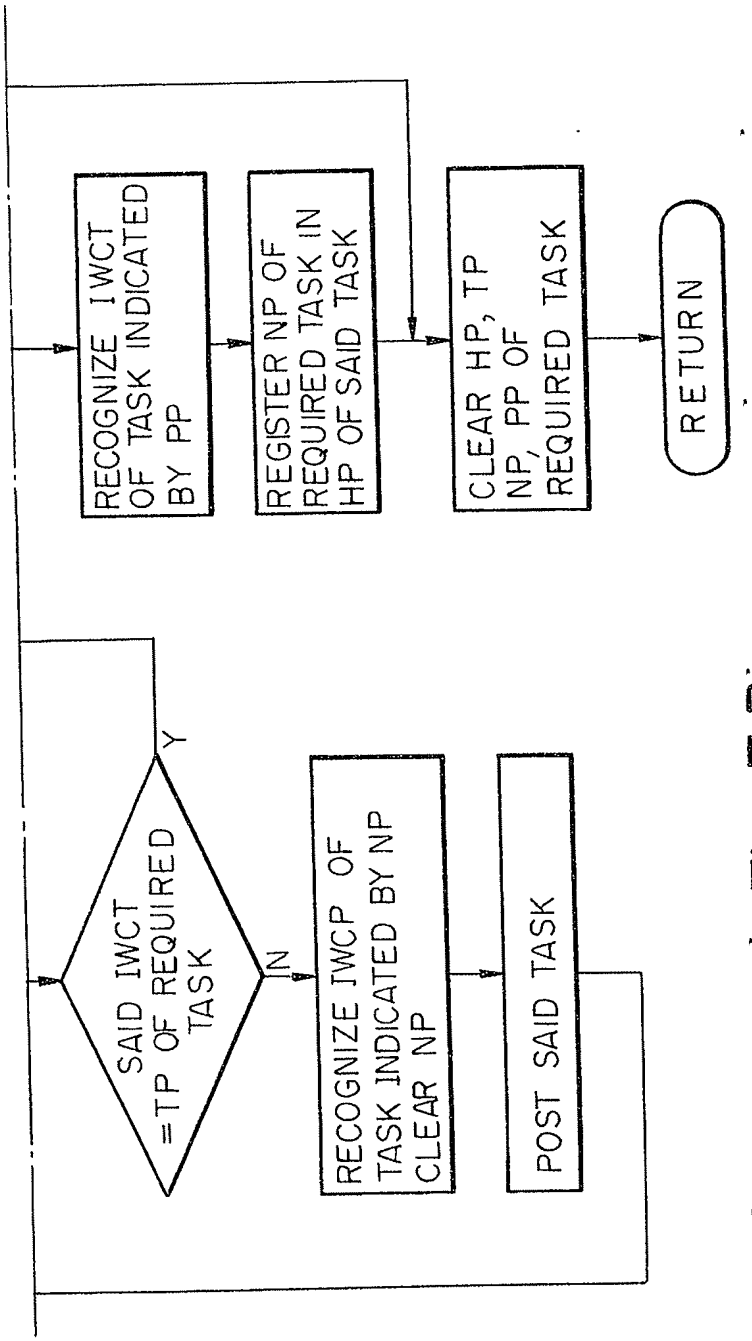


Fig. 7B

Mr Judder Jordan + Co.
Patent Agents